Search Results

Search Results for: **[phi<AND>(("static single assignment"))]**
Found **40** of **131,734 searched.**

Search within Results

> Advanced Search

> Search Help/Tips

Sort by: Title Publication Publication Date Score Binder

Results 1 - 20 of 40 short listing

Prev Page 1 2 3 Next Page

1 SafeTSA: a type safe and referentially secure mobile-code representation based on static single assignment form 94%
Wolfram Amme , Niall Dalton , Jeffery von Ronne , Michael Franz
ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2001 conference on Programming language design and implementation May 2001
Volume 36 Issue 5

2 A new algorithm for scalar register promotion based on SSA form 90%
A. V. S. Sastry , Roy D. C. Ju
ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1998 conference on Programming language design and implementation May 1998
Volume 33 Issue 5

We present a new register promotion algorithm based on Static Single Assignment (SSA) form. Register promotion is aimed at promoting program names from memory locations to registers. Our algorithm is profile-driven and is based on the scope of intervals. In cases where a complete promotion is not possible because of the presence of function calls or pointer references, the proposed algorithm is capable of eliminating loads and stores on frequently executed paths by placing loads and stores on le ...

3 Compilation techniques for reconfigurable devices: Data communication estimation and reduction for reconfigurable systems 89%
Adam Kaplan , Philip Brisk , Ryan Kastner
Proceedings of the 40th conference on Design automation June 2003

Widespread adoption of reconfigurable devices requires system level synthesis techniques to take an application written in a high level language and map it to the reconfigurable device. This paper describes methods for synthesizing the internal representation of a compiler into a hardware description language in order to program

reconfigurable hardware devices. We demonstrate the usefulness of static single assignment (SSA) in reducing the amount of data communication in the hardware. However, t ...

4 Algorithms for computing the static single assignment form 87%

Gianfranco Bilardi , Keshav Pingali
Journal of the ACM (JACM) May 2003
Volume 50 Issue 3

The Static Single Assignment (SSA) form is a program representation used in many optimizing compilers. The key step in converting a program to SSA form is called φ-placement. Many algorithms for φ-placement have been proposed in the literature, but the relationships between these algorithms are not well understood.In this article, we propose a framework within which we systematically derive (i) properties of the SSA form and (ii) φ-placement algorithms. This framework is based on a n ...

5 Static single assignment form for machine code 87%

Allen Leung , Lal George
ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1999 conference on Programming language design and implementation May 1999
Volume 34 Issue 5

Static Single Assignment (SSA) is an effective intermediate representation in optimizing compilers. However, traditional SSA form and optimizations are not applicable to programs represented as native machine instructions because the use of dedicated registers imposed by calling conventions, the runtime system, and target architecture must be made explicit. We present a simple scheme for converting between programs in machine code and in SSA, such that references to dedicated physical registers ...

6 A new algorithm for partial redundancy elimination based on SSA form 85%

Fred Chow , Sun Chan , Robert Kennedy , Shin-Ming Liu , Raymond Lo , Peng Tu
ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1997 conference on Programming language design and implementation May 1997
Volume 32 Issue 5

A new algorithm, SSAPRE, for performing partial redundancy elimination based entirely on SSA form is presented. It achieves optimal code motion similar to lazy code motion [KRS94a, DS93], but is formulated independently and does not involve iterative data flow analysis and bit vectors in its solution. It not only exhibits the characteristics common to other sparse approaches, but also inherits the advantages shared by other SSA-based optimization techniques. SSAPRE also maintains its output in t ...

7 Code scheduling: Phi-Predication for light-weight if-conversion 84%

Weihaw Chuang , Brad Calder , Jeanne Ferrante
Proceedings of the international symposium on Code generation and optimization: feedback-directed and runtime optimization March 2003

Predicated execution can eliminate hard to predict branches and help to enable instruction level parallelism. Many current predication variants exist where the result update is conditional based upon the outcome of the guarding predicate. However, conditional writing of a register creates a naming problem for an out-of-order processor, and can stall the issuing of instructions. This problem arises from potential multiple predicated definitions reaching a use, which is unresolved until the prior ...

8 A schedular-sensitive global register allocator 84%

C. Norris , L. L. Pollock

Proceedings of the 1993 ACM/IEEE conference on Supercomputing December 1993

9 Static memory allocation by pointer analysis and coloring 82%

J. Zhu

Proceedings of the conference on Design, automation and test in Europe March 2001

10 Analysis of pointers and structures 82%

David R. Chase , Mark Wegman , F. Kenneth Zadeck

ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1990 conference on Programming language design and implementation June 1990
Volume 25 Issue 6

11 LLVA: A Low-level Virtual Instruction Set Architecture 82%

Vikram Adve , Chris Lattner , Michael Brukman , Anand Shukla , Brian Gaeke

Proceedings of the 36th Annual IEEE/ACM International Symposium on Microarchitecture December 2003

A virtual instruction set architecture (V-ISA) implementedvia a processor-specific software translation layercan provide great flexibility to processor designers. Recentexamples such as Crusoe and DAISY, however, haveused existing hardware instruction sets as virtual ISAs,which complicates translation and optimization. In fact,there has been little research on specific designs for a virtualISA for processors. This paper proposes a novel virtualISA (LLVA) and a translation strategy for implementi ...

12 Intermediate representation engineering: Efficient online optimization by utilizing offline analysis and the safeTSA representation 82%

Jeffery von Ronne , Andreas Hartmann , Wolfram Amme , Michael Franz

Proceedings of the inaugural conference on the Principles and Practice of programming, 2002 and Proceedings of the second workshop on Intermediate representation engineering for virtual machines, 2002 June 2002

Conventional mobile-code representations, e.g. Java bytecode, provide machine-independence and type-safety, but do so at the expense of performance. This performance hit can be taken in the form of *decreased throughput or in increased latency*. SafeTSA was designed to reduce this performance hit, especially when producing high-quality optimized machine code. It does this by utilizing SSA form and thus shifting dataflow analysis effort from the online JIT compiler to the offline producer of ...

13 Interprocedural compatibility analysis for static object preallocation 82%

Ovidiu Gheorghioiu , Alexandru Salcianu , Martin Rinard

ACM SIGPLAN Notices , Proceedings of the 30th ACM SIGPLAN-SIGACT symposium on Principles of programming languages January 2003
Volume 38 Issue 1

We present an interprocedural and compositional algorithm for finding pairs of *compatible* allocation sites, which have the property that no object allocated at one site is live at the same time as any object allocated at the other site. If an allocation site is compatible with itself, it is said to be *unitary*: at most one object allocated at that site is live at any given point in the, execution of the program. We use the results of the analysis to statically preallocate memory spa ...

14 Partial redundancy elimination in SSA form 82%

Robert Kennedy , Sun Chan , Shin-Ming Liu , Raymond Lo , Peng Tu , Fred Chow
ACM Transactions on Programming Languages and Systems (TOPLAS) May 1999
Volume 21 Issue 3

The SSAPRE algorithm for performing partial redundancy elimination based entirely on SSA form is presented. The algorithm is formulated based on a new conceptual framework, the factored redundancy graph, for analyzing redundancy, and representes the first sparse approach to the classical problem and on methods for its solution. With the algorithm description, theorems and their proofs are given showing that the algorithm produces the best possible code by the criteria of computational optim ...

15 Global code motion/global value numbering 82%

Cliff Click
ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1995 conference on Programming language design and implementation June 1995
Volume 30 Issue 6

16 Compilation: Efficient static single assignment form for predication 82%

Arthur Stoutchinin , Francois de Ferriere
Proceedings of the 34th annual ACM/IEEE international symposium on Microarchitecture December 2001

We present a framework that allows translation of predicated code into the static single assignment (SSA) form, and simplifies application of the SSA-based optimizations to predicated code. In particular, we represent predicate join points in the program by the Ψ-functions similar to the Φ-functions of the basic SSA. The SSA-based optimizations (such as constant propagation) can be applied to predicated code by simply specifying additional rules for processing the Ψ-functions. We pre ...

17 A compiler framework for speculative analysis and optimizations 80%

Jin Lin , Tong Chen , Wei-Chung Hsu , Pen-Chung Yew , Roy Dz-Ching Ju , Tin-Fook Ngai , Sun Chan
ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2003 conference on Programming language design and implementation May 2003
Volume 38 Issue 5

Speculative execution, such as control speculation and data speculation, is an effective way to improve program performance. Using edge/path profile information or simple heuristic rules, existing compiler frameworks can adequately incorporate and exploit control speculation. However, very little has been done so far to allow existing compiler frameworks to incorporate and exploit data speculation effectively in various program transformations beyond instruction scheduling. This paper proposes a ...

18 EPIC compilation: Speculative register promotion using Advanced Load Address Table (ALAT) 80%

Jin Lin , Tong Chen , Wei-Chung Hsu , Pen-Chung Yew
Proceedings of the international symposium on Code generation and optimization: feedback-directed and runtime optimization March 2003

The pervasive use of pointers with complicated patterns in C programs often constrains compiler alias analysis to yield conservative register allocation and promotion. Speculative register promotion with hardware support has the potential to more aggressively promote memory references into registers in the presence of aliases. This paper studies the use of the Advanced Load Address Table (ALAT), a data speculation feature defined in the IA-64 architecture, for speculative register promotion. An ...

19 Intermediate representation engineering: Improving mobile program performance through the use of a hybrid intermediate representation 80%

Chandra Krintz

Proceedings of the inaugural conference on the Principles and Practice of programming, 2002 and Proceedings of the second workshop on Intermediate representation engineering for virtual machines, 2002 June 2002

We present a novel transfer format for mobile programs that is a hybrid of two existing formats: bytecode and SafeTSA. Java bytecode offers a compact representation and ease of interpretation (fast-compilation); SafeTSA offers amenability to optimization. We use program profiling to guide format selection at the method-level. Methods deemed "hot" are those for which optimization should be expended and as such, are encoded using the SafeTSA format. All other methods are encoded using bytecode. Ou ...

20 Architecture 2: Using predicate path information in hardware to determine true dependences 80%

Lori Carter , Brad Calder

Proceedings of the 16th international conference on Supercomputing June 2002

Predicated Execution has been put forth as a method for improving processor performance by removing hard-to-predict branches. As part of the process of turning a set of basic blocks into a predicated region, both paths of a branch are combined into a single path. There can be multiple definitions from disjoint paths that reach a use. Waiting to find out the correct definition that actually reaches the use can cause pipeline stalls.In this paper we examine a hardware optimization that dynamically ...

Results 1 - 20 of 40 short listing

Prev Page 1 2 3 Next Page